Building Xsan Applications

Session 622

These are confidential sessions—please refrain from filming or taking pictures.

Enterprise IT

Building Xsan Applications

Jason Thorpe
Xsan Engineering Manager

http://developer.apple.com/wwdc2005/

Xsan: The SAN File System for Mac OS X

What You'll Learn

- How Xsan works
- Xsan volume characteristics
- Tips on configuring your Xsan volumes
- Tips on making your applications Xsan compatible
- Tips on improving application performance with Xsan
- Xsan developer APIs
- Setting up an Xsan sandbox for application development
- Xsan Admin

How Xsan Works

http://developer.apple.com/wwdc2005/

File System Types

Traditional file systems

File System Types

Traditional file systems

- Direct Attach Storage
  - Single disk contains catalog and data
  - File system code runs on single computer
  - Examples: HFS+, UFS

File System Types

Traditional file systems

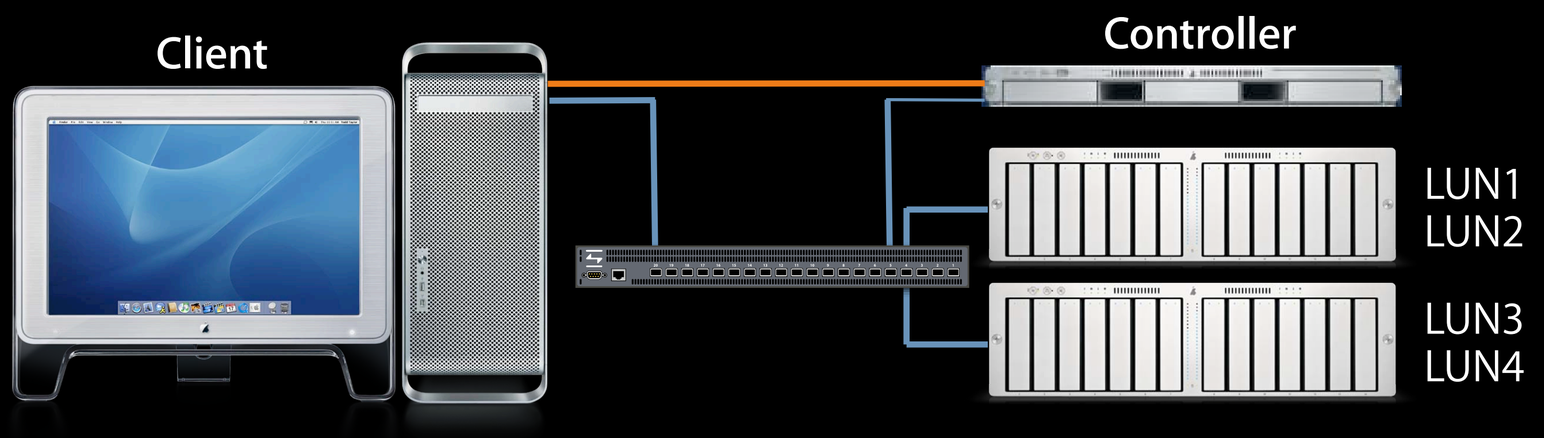
- Direct Attach Storage
  - Single disk contains catalog and data
  - File system code runs on single computer
  - Examples: HFS+, UFS
- Network Attached Storage
  - Allows multiple computers to share files
  - Accepts high-level file requests from clients
  - Server translates calls to local file system
  - Examples: AFP, NFS

File System Types

SAN file system

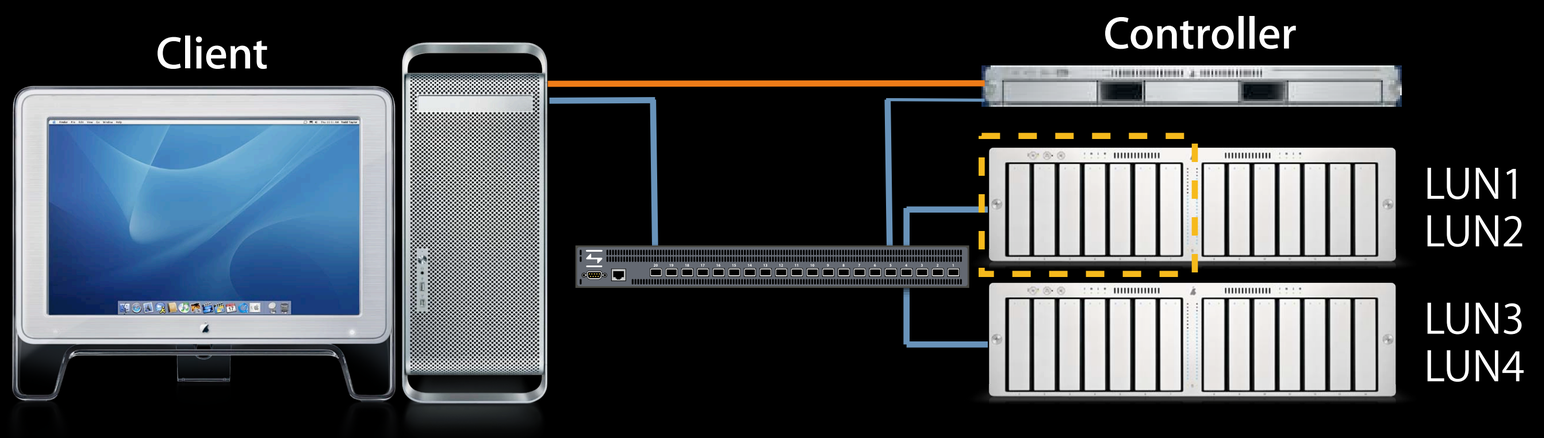
- Disks are attached to all computers on the SAN
- Catalog and data are stored separately
- One process reads catalog data and acts as a server to translate file offsets to disk offsets
- File system on each client contacts the server for catalog requests but does I/O directly to the disks

How Xsan Works



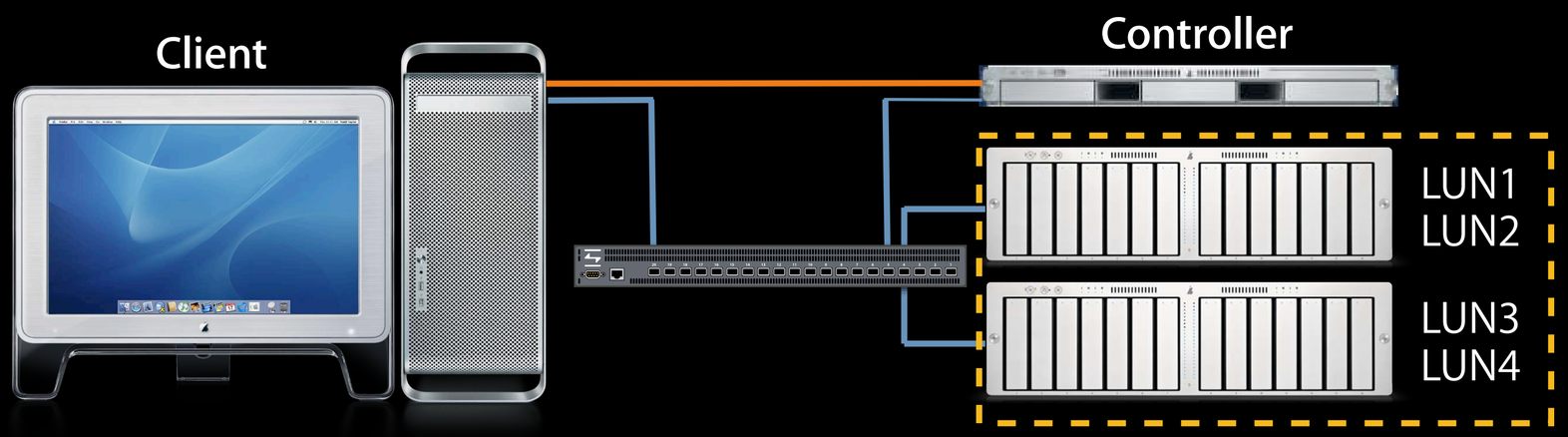
- Example Xsan setup
  - Systems hooked together with Fibre Channel
  - Clients connected to controller on private IP network
  - 4 Xserve-RAID LUNs
    - LUN1 – RAID 1
    - LUN2, LUN3, LUN4 – RAID 5

How Xsan Works



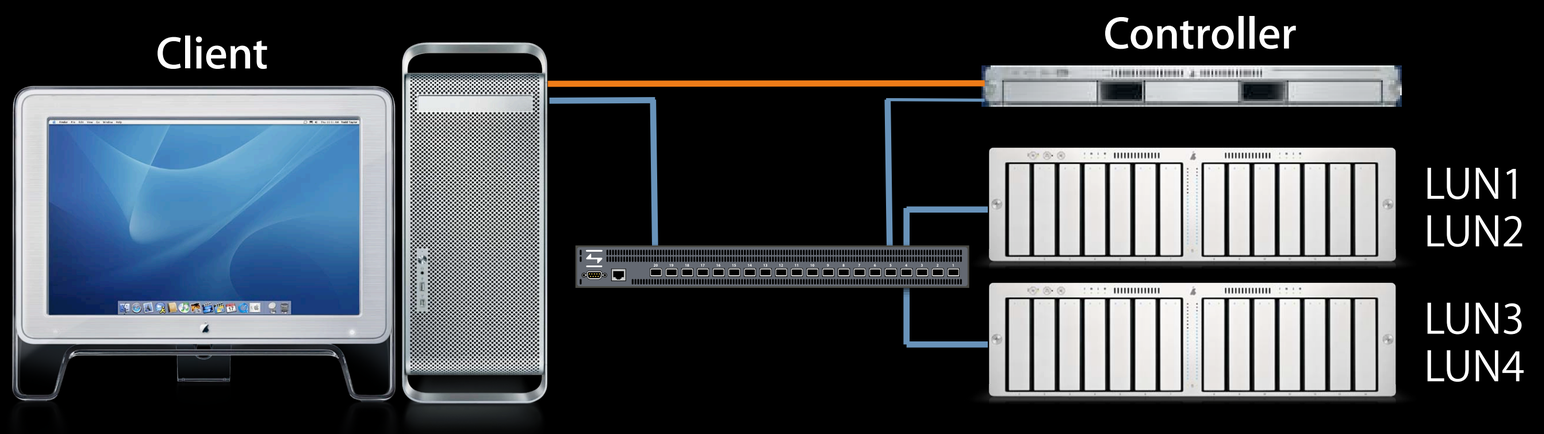
- One LUN is selected to store metadata

How Xsan Works



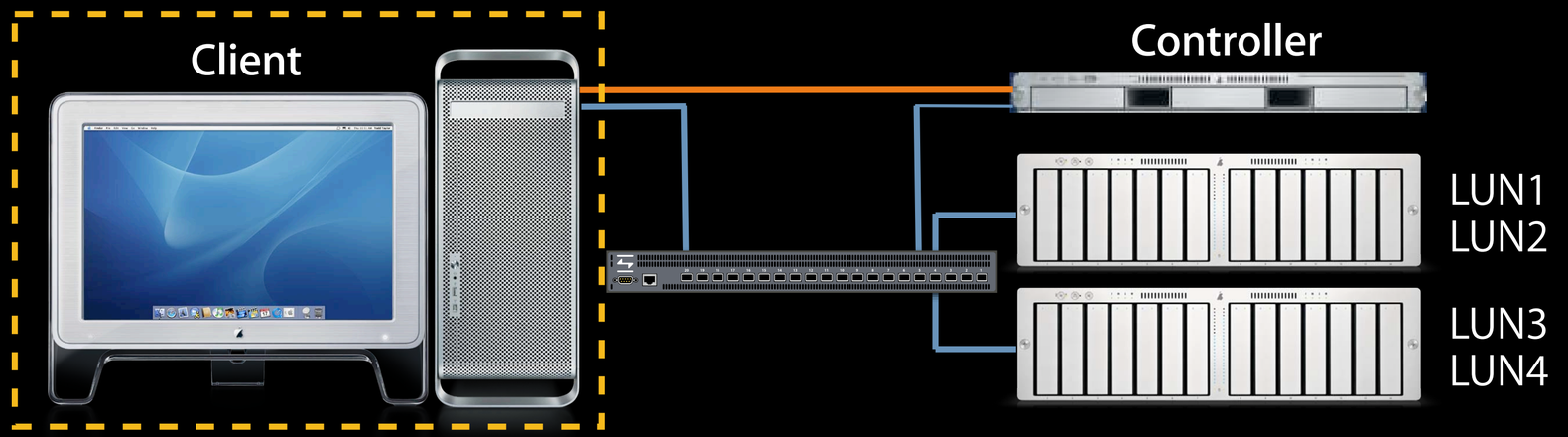
- One LUN is selected to store metadata
- Controller is configured to know which LUNs are to be grouped into a volume

How Xsan Works



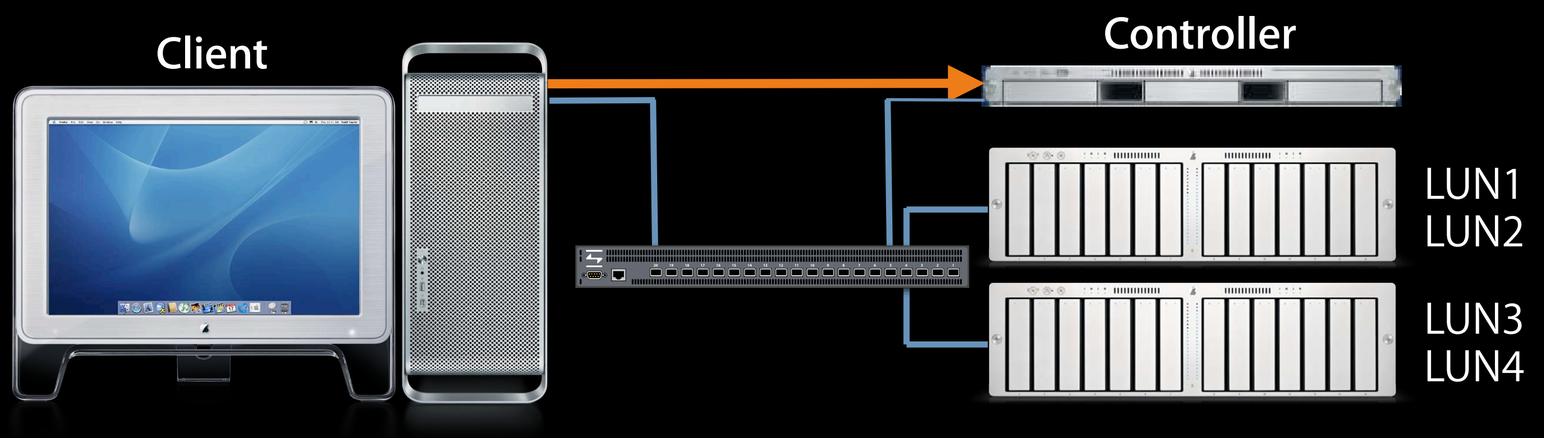
- One LUN is selected to store metadata
- Controller is configured to know which LUNs are to be grouped into a volume
- Client-side file system has no knowledge of the file system layout

How Xsan Works



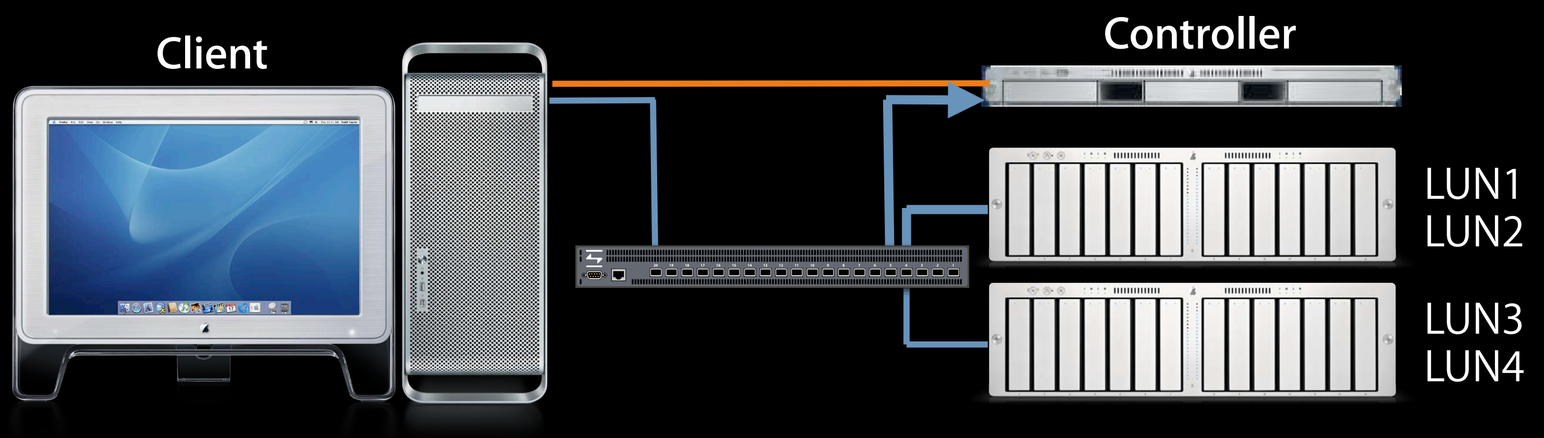
- Application makes “write” call to Xsan volume
  - write 12K to file at offset 0

How Xsan Works



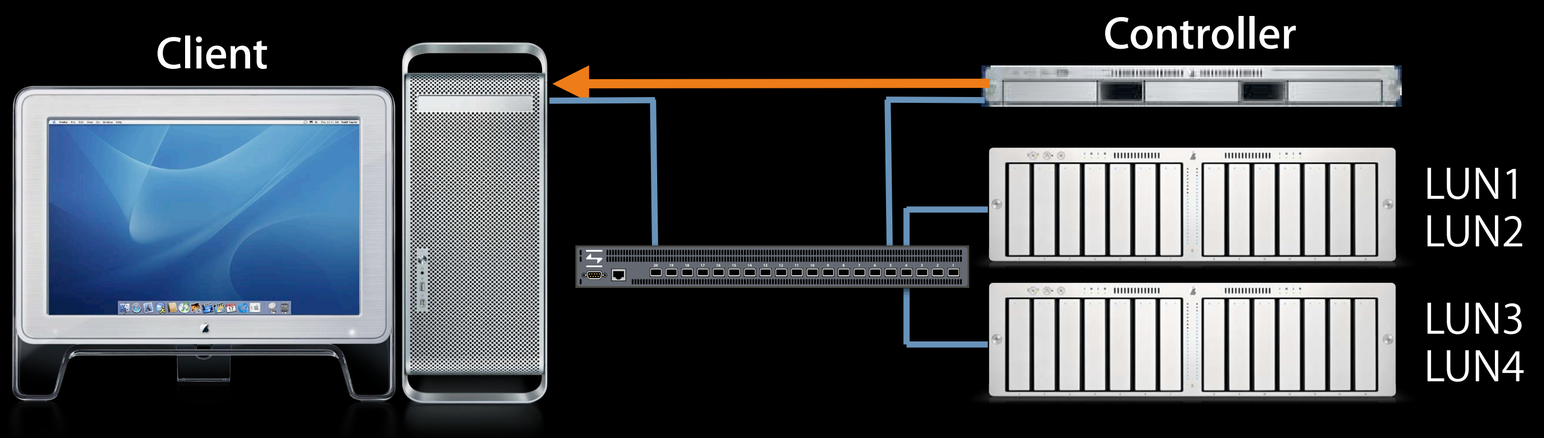
- Application makes “write” call to Xsan volume
- Xsan file system makes call to controller
  - Where is the data for file (offset 0 to 12K) stored?

How Xsan Works



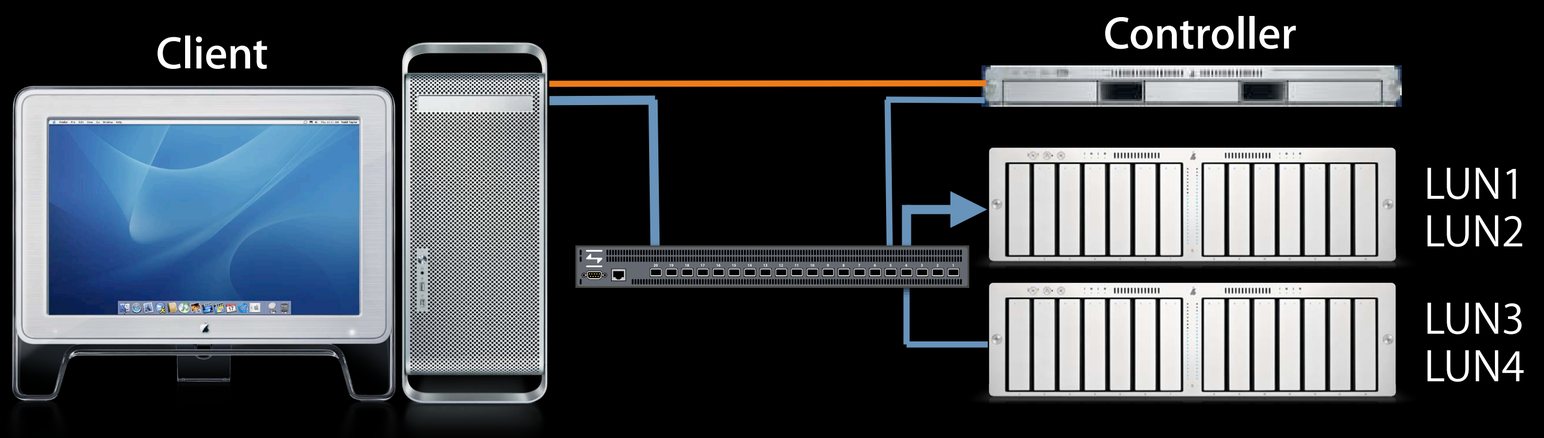
- Application makes “write” call to Xsan volume
- Xsan file system makes call to controller
- Controller reads catalog from LUN1

How Xsan Works



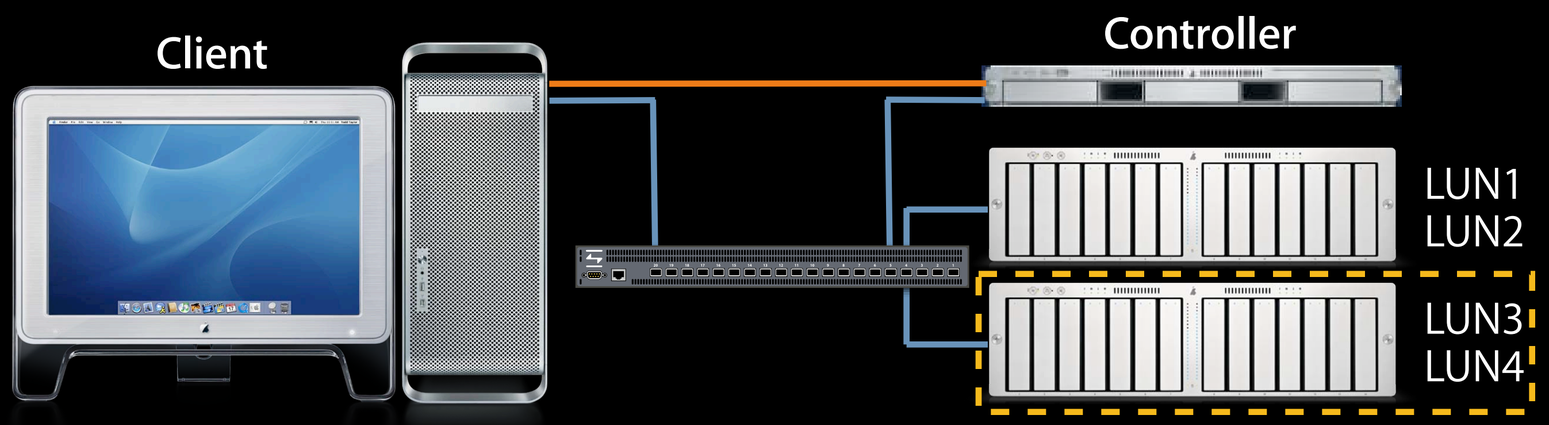
- Application makes “write” call to Xsan volume
- Xsan file system makes call to controller
- Controller reads catalog from LUN1
- Controller tells client where file lives (file extents)
  - File (0-12K) is on LUN2 at offset 100

How Xsan Works



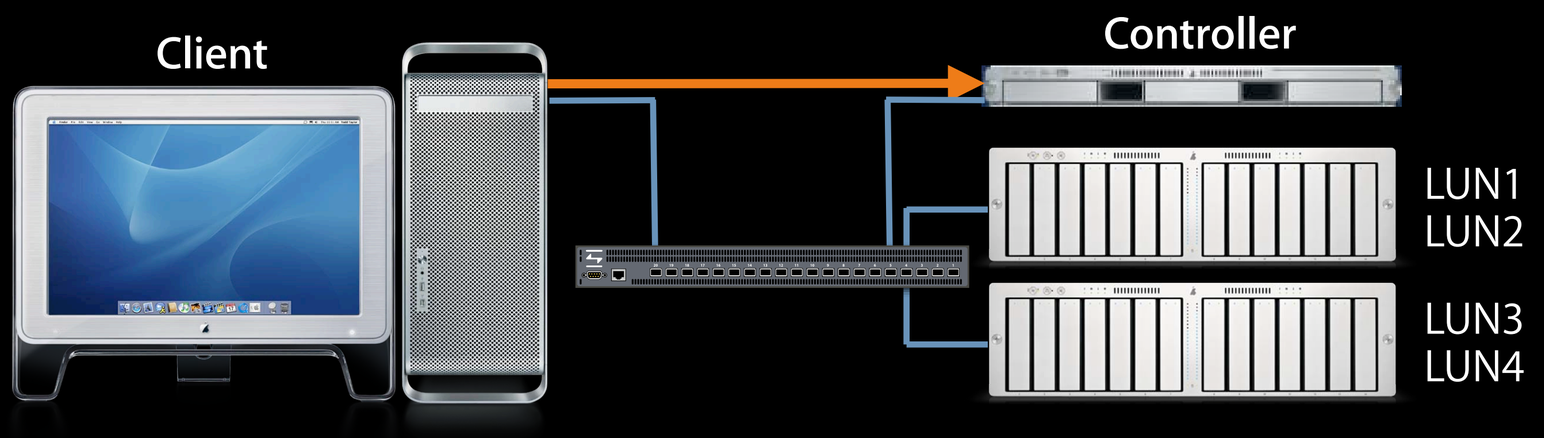
- Application makes “write” call to Xsan volume
- Xsan file system makes call to controller
- Controller reads catalog from LUN1
- Controller tells client where file lives (file extents)
- Client writes data directly to disk
  - Write to LUN2 at offset 100

How Xsan Works



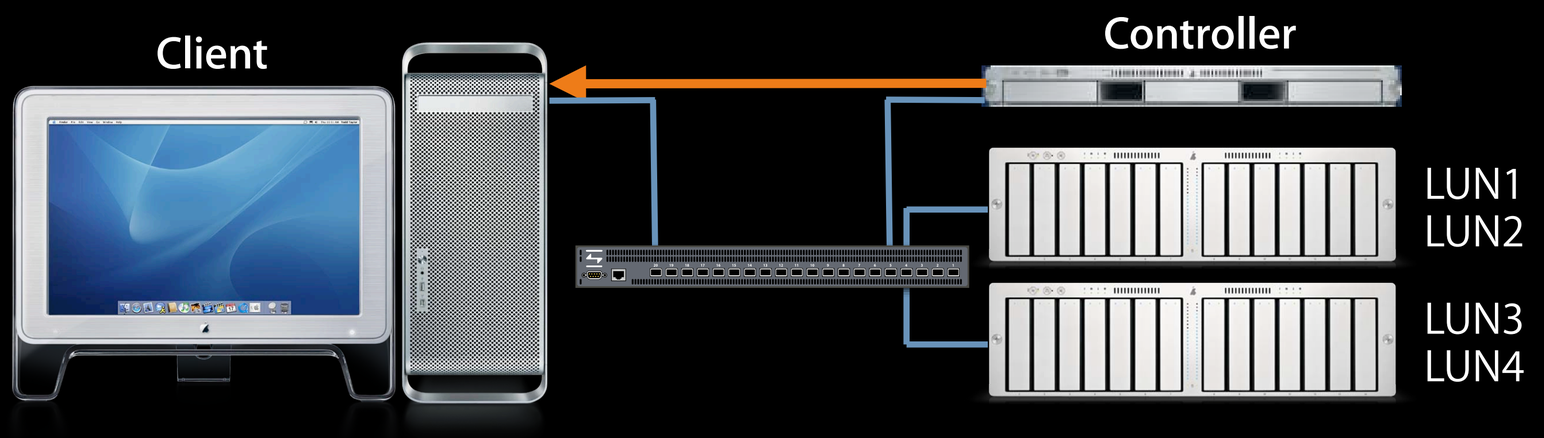
- Storage Pools group LUNs for better performance
  - Similar result to software RAID 0
  - Handled by controller, not a driver

How Xsan Works



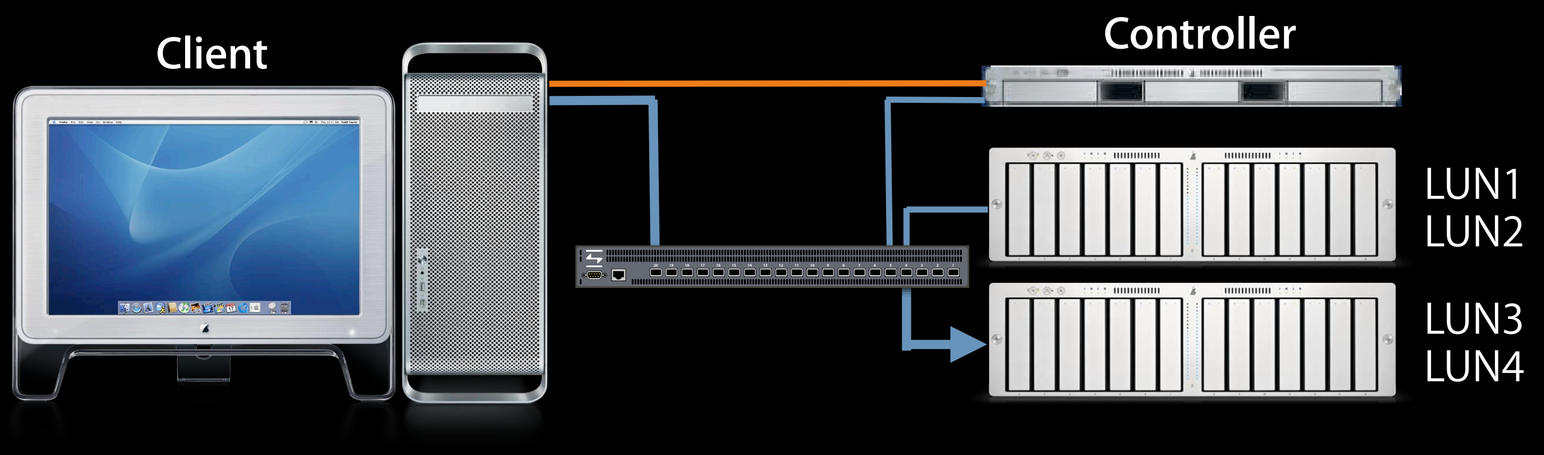
- Client makes same request to controller
  - Where is the data for file (offset 0 to 12K) stored?

How Xsan Works



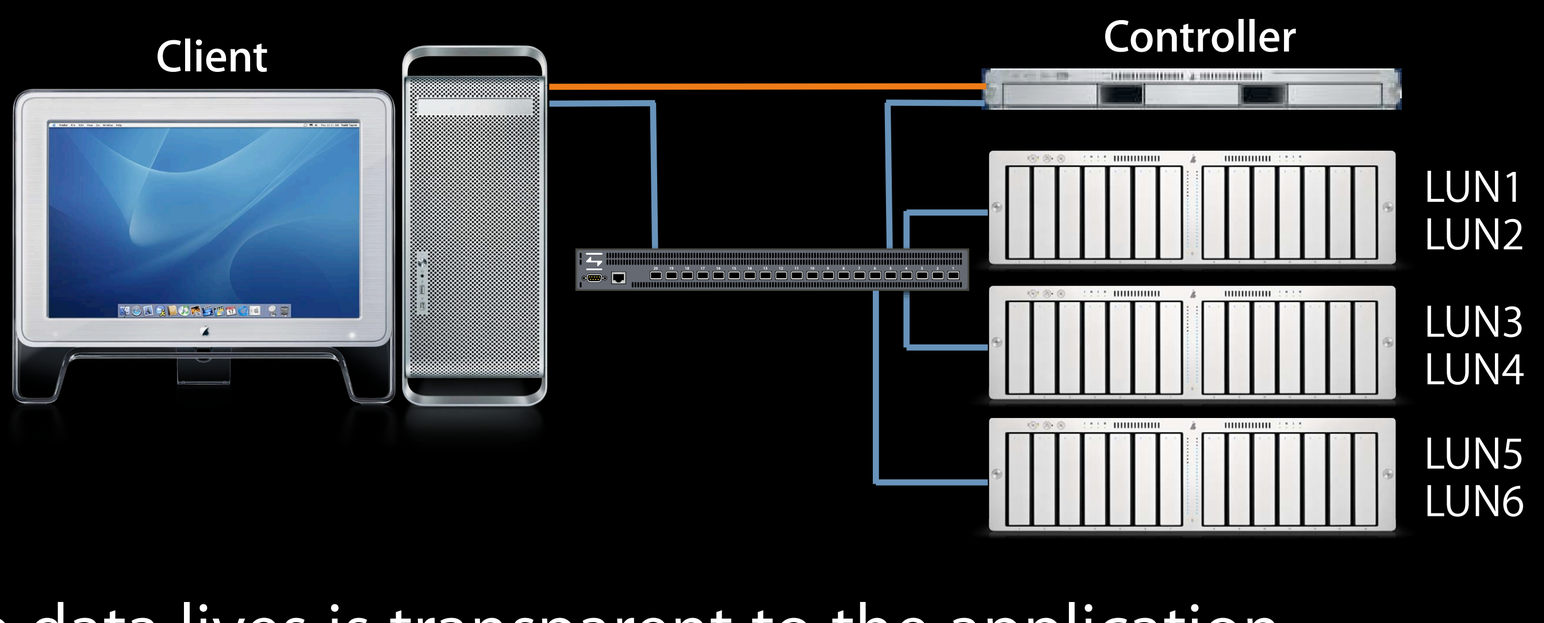
- Client makes same request to controller
  - Where is the data for file (offset 0 to 12K) stored?
- Controller responds
  - File 0-6K on LUN3 at offset 100, 6-12K on LUN4 at offset 100

How Xsan Works



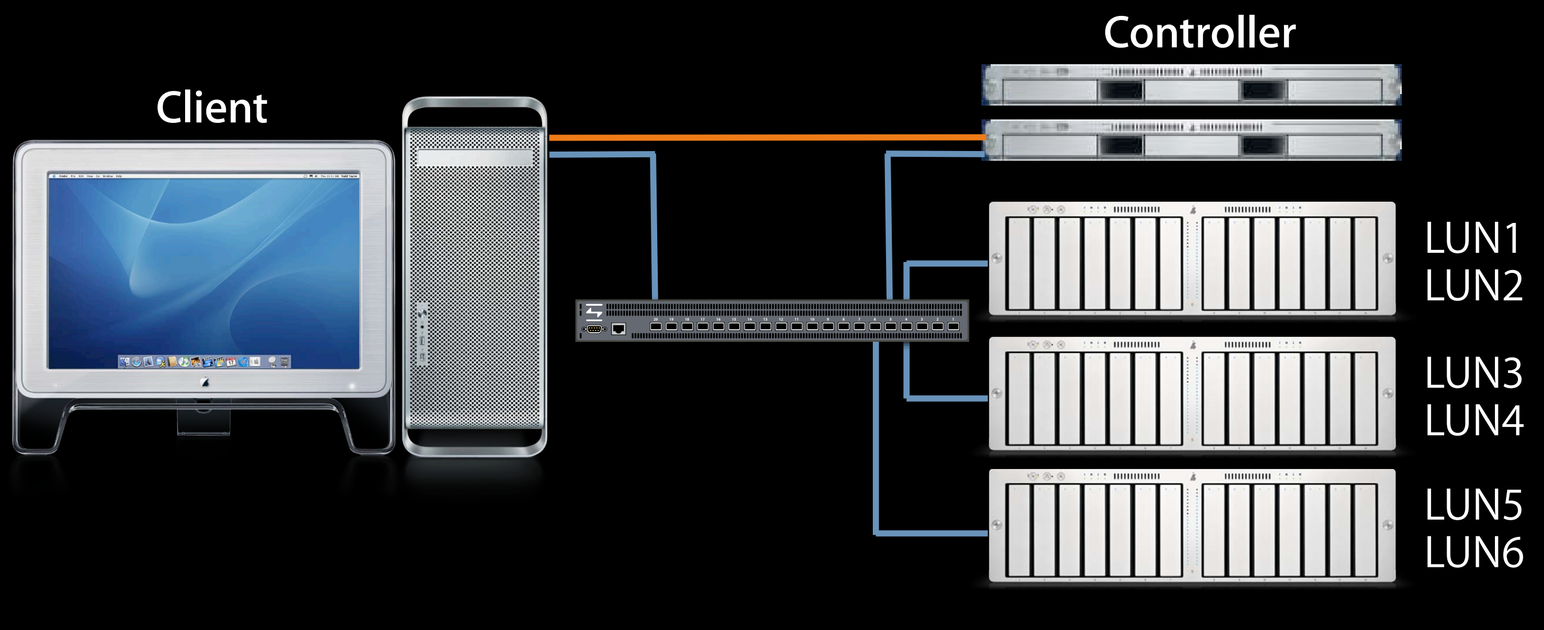
- Client makes same request to controller
  - Where is the data for file (offset 0 to 12K) stored?
- Controller responds
  - File 0–6K on LUN3 at offset 100, 6–12K on LUN4 at offset 100
- Client writes data to both LUN3 and LUN4 in parallel

How Xsan Works



- Where data lives is transparent to the application
- Can add more storage by reconfiguring controller

How Xsan Works



- If primary controller fails, backup can take over
  - Uses journal data to reconstruct file system state
  - Clients reestablish locks
  - Controller can be up and running in seconds
    - During this time, clients can still stream data

Xsan Configuration Tips

http://developer.apple.com/wwdc2005/

Xsan Configuration Tips

Metadata storage

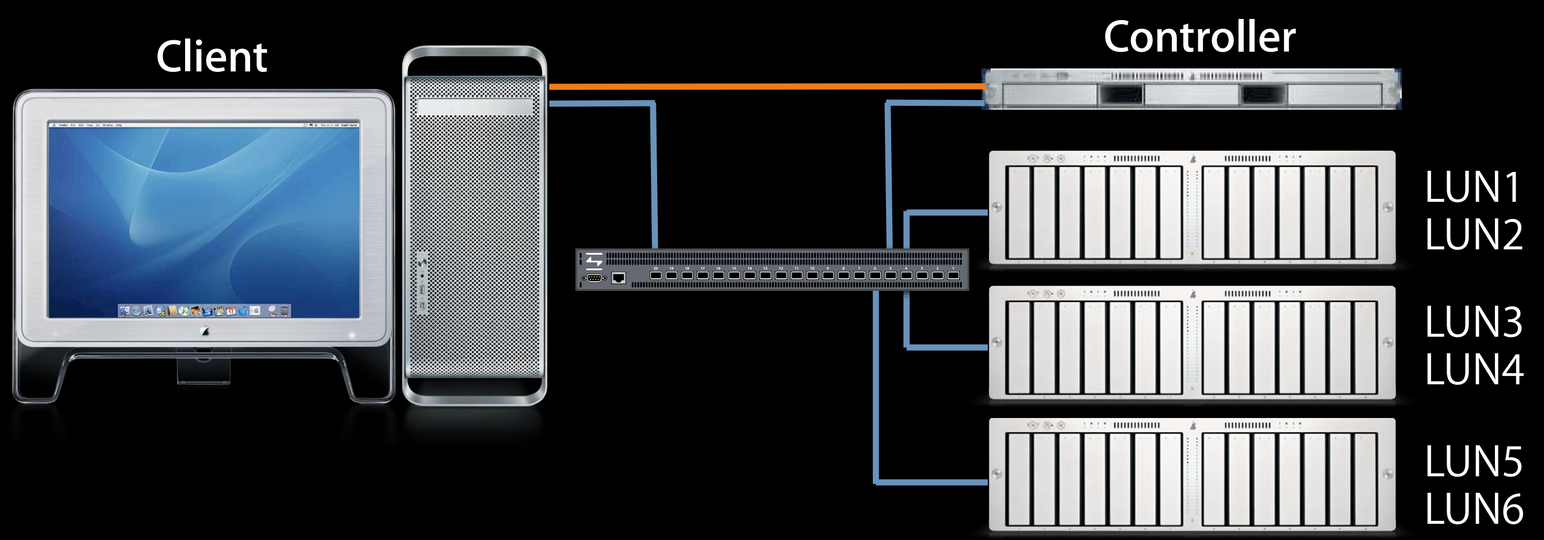
- Metadata is comprised of catalog information and journal
- Should be isolated from other data
- LUN should be redundant with hot spare
- Good random I/O performance
- Needs approximately 10G per 10 million files

Xsan Configuration Tips

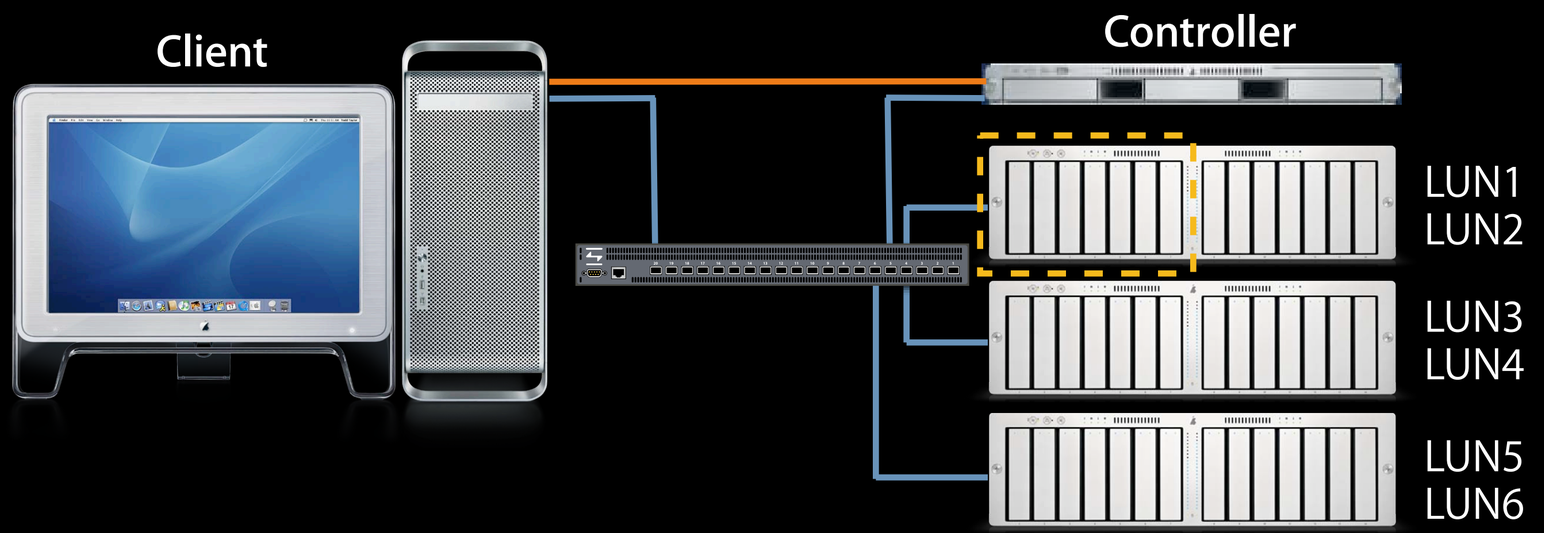
Data storage

- Combine LUNs into storage pools
  - Increases performance through striping
  - Leverages Fibre Channel multi-pathing
  - Only combine LUNs with same characteristics
- Create different storage pools for different workloads
  - Choose the optimal RAID configuration for each type of data
  - I/O on one storage pool won’t interfere with I/O on another

Storage Pool Example

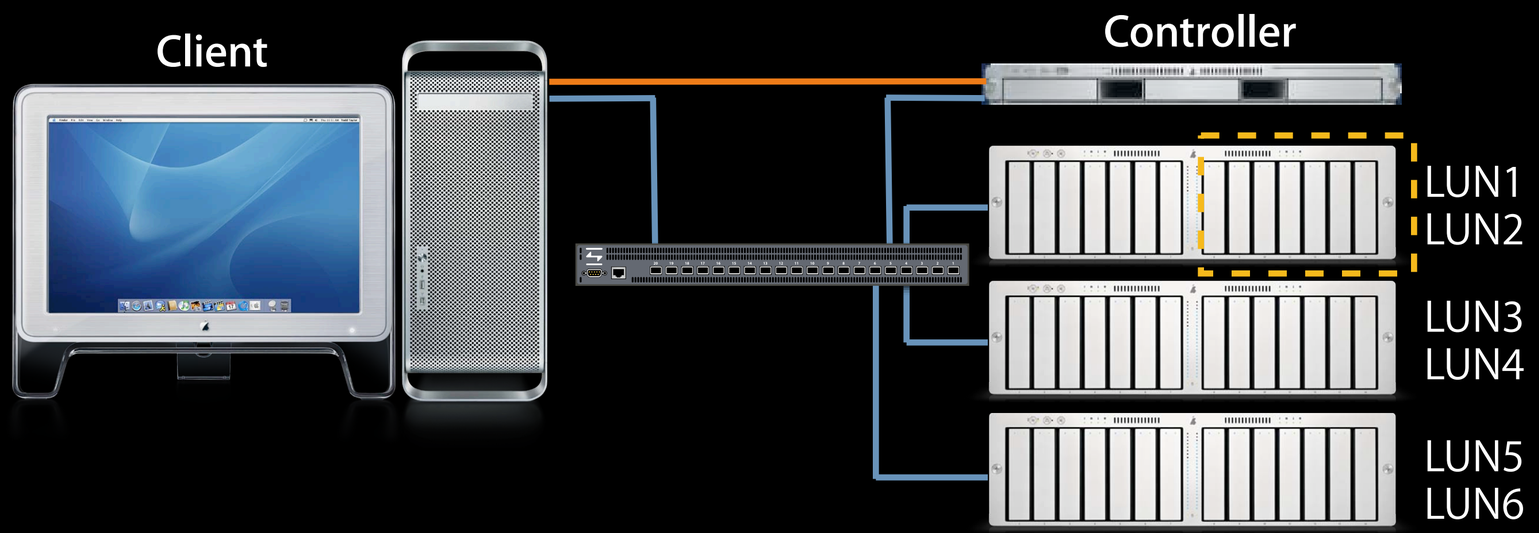


Storage Pool Example



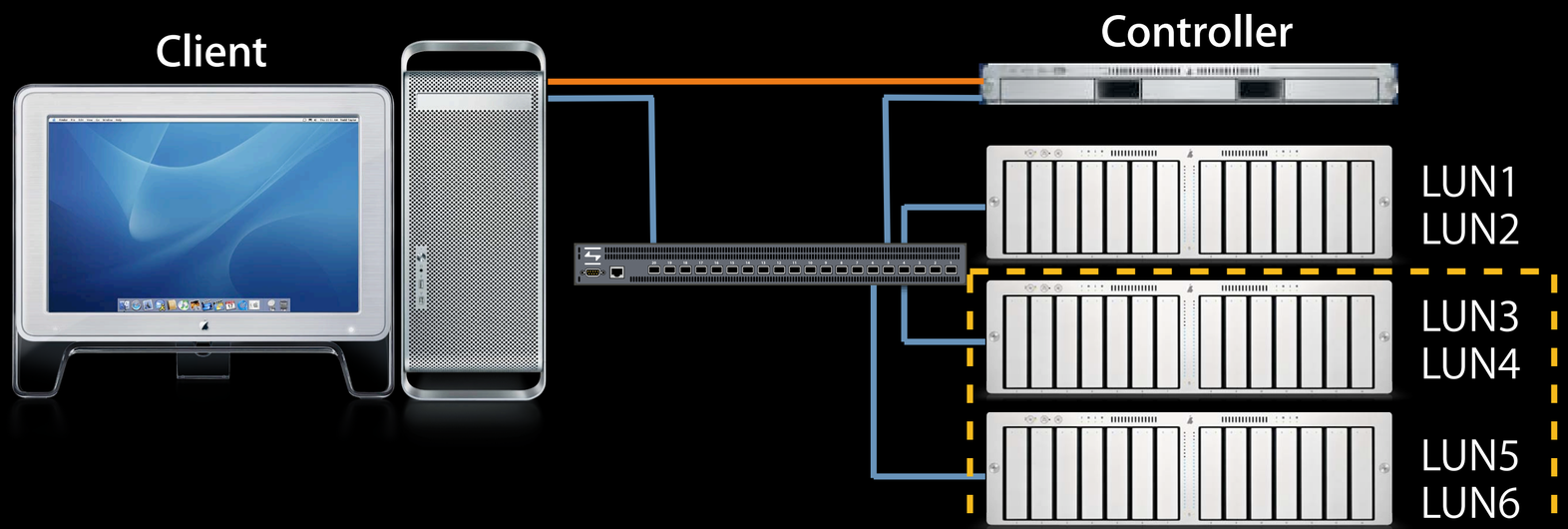
- 1 metadata storage pool (1x RAID 1)

Storage Pool Example



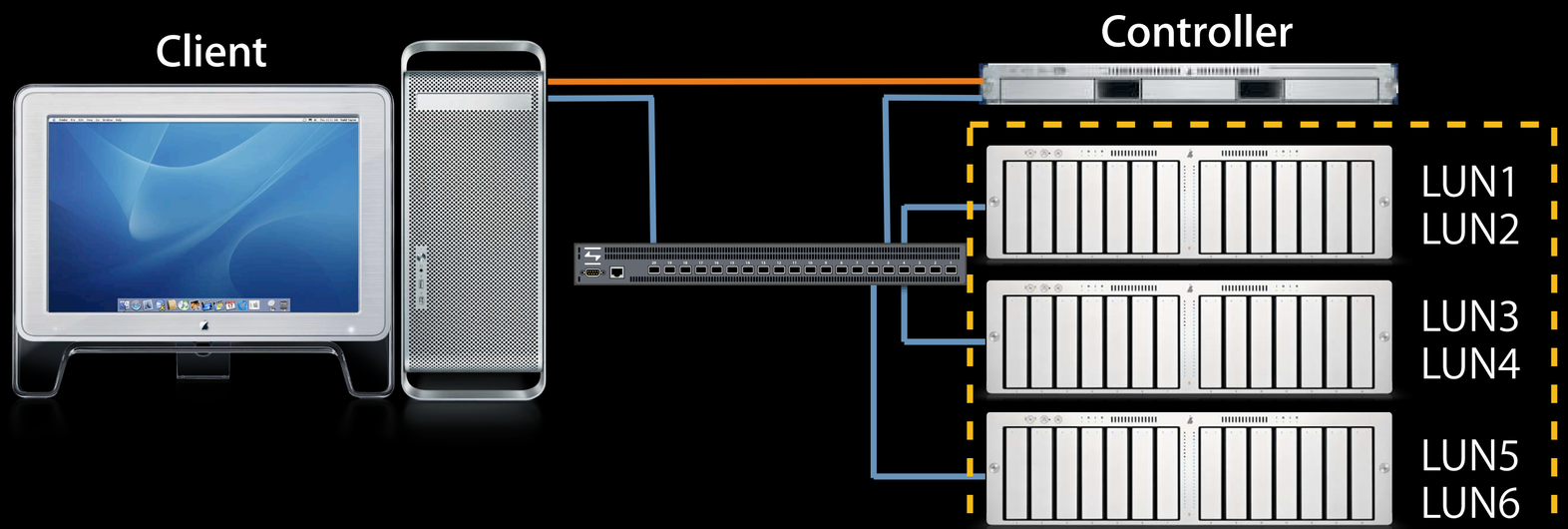
- 1 metadata storage pool (1x RAID 1)
- 1 audio storage pool (1x RAID 5)

Storage Pool Example



- 1 metadata storage pool (1x RAID 1)
- 1 audio storage pool (1x RAID 5)
- 1 video storage pool (4x RAID 5)

Storage Pool Example



- 1 metadata storage pool (1x RAID 1)
- 1 audio storage pool (1x RAID 5)
- 1 video storage pool (4x RAID 5)
- All grouped into a single volume!

Affinities

- Controller usually assigns files to storage pools in “round robin” fashion
- Affinities steer file data to specific storage pools
- Storage pools can be marked as “exclusive”
- Xsan Admin allows mapping folders to specific storage pools
- Command line tool
- Application API

Xsan Volume Characteristics

What to expect, and how to deal with it

http://developer.apple.com/wwdc2005/

Performance Characteristics

- Metadata operations faster than traditional file server
  - IP network not flooded with I/O
  - Private network should be lightly loaded and highly responsive
  - Controller only handles metadata, not I/O operations
- However, metadata still slower than local volume
  - Lookup, create, delete, etc. all require round-trips to controller
- File I/O at RAID speed

Xsan Compatibility Tips

- Shared volume, need to do file locking as if on a network volume
  - Supports open() flags (O_SHLOCK and O_EXLOCK) as well as F_SETLK fcntl for byte range locking
- Very big volumes
  - Multi-terabyte volumes will be common
  - Larger numbers of files as well as larger file sizes
- Not an HFS+ volume
  - Case-sensitive
  - Carbon uses AppleDouble
- Be aware of file ownership and permissions
  - No user ID mapping
  - Uses standard POSIX uid/gid and permissions

Xsan Performance Tips

- Reduce metadata operations
  - Try to avoid using lots of small files
  - Use Xsan APIs to preload file extents when streaming data
- Keep file fragmentation to a minimum
  - Preallocate storage when writing files
    - Supports F_PREALLOCATE fcntl (POSIX) and FSAllocateFork() (Carbon)
- Issue large I/O requests when possible
  - Helps file system avoid fragmentation
  - Allows file system to exploit parallelism

Xsan Developer APIs

http://developer.apple.com/wwdc2005/

Xsan Developer APIs

- Provides access to Xsan-specific features
  - Extent preloading
  - Bandwidth reservation
  - Affinities
- Underlying API uses sysctl()
  - Header file and sample code using this API on Xsan CD
- API still in flux
  - Effort will be made to maintain binary compatibility
  - Working toward better integration with standard Mac OS X APIs

Preallocating Files—POSIX

- Use fcntl()

```
fstore_t fst;

fst.fst_flags = 0;
fst.fst_posmode = F_PEOFPOSMODE;
fst.fst_offset = 0;
fst.fst_length = file_size;
fst.fst_bytesalloc = 0;  /* OUT: # of bytes allocated */

status = fcntl(fd, F_PREALLOCATE, &fst);

if (status != 0)
{
  printf("F_PREALLOCATE call failed\n");
  return 0;
}
```

Preallocating Files—Carbon

- Use FSAllocateFork()

```
UInt64 bytesAllocated;
OSErr error;

error = FSAllocateFork(ref, kFSAllocDefaultFlags,
    fsFromLEOF, 0, file_size, &bytesAllocated);

if (error != 0)
{
  printf("FSAllocateFork() call failed\n");
  return 0;
}
```

Determining Volume Type

- Use statfs()

```
status = statfs(path, &buf);

if (status != 0)
{
  printf("statfs call failed\n");
  return 0;
}
else if (strcmp(buf.f_fstypename, XSANFSTYPENAME) != 0)
{
  printf("Path does not point to an Xsan filesystem\n");
  return 0;
}
```

Making an Xsan API call

- You can get the Xsan version with F_GETVERINFO API call

```
typedef struct _verinfo_reply {
    char        vr_version[CVVERINFO_MAX];
    char        vr_build[CVVERINFO_MAX];
    char        vr_creationdate[CVVERINFO_MAX];
    uint32_t    vr_apiversion;
    uint32_t    vr_pad1;
} VerInfoReply_t;
```

```
fd = open(argv[1], O_RDONLY, 0);
status = FsSysCtl(fd, F_GETVERINFO, &reply);

if (status == 0)
  printf("Version = %s\n", reply.vr_version);
```

Preloading File Extents

- Use the F_LOADEXT API call

```
LoadExtReq_t lq;

lq.lq_size = file_size; /* from fstat() call */
lq.lq_offset = 0;

status = FsSysCtl(fd, F_LOADEXT, &lq);

if (status != 0)
{
  printf("F_LOADEXT call failed\n");
  return 0;
}
```

Bandwidth Reservation

- Use F_SETRTIO API call to enable real-time I/O on a file
- Requires additional configuration
  - Edit /Library/Filesystems/Xsan/config/*volumename*.cfg
  - Specify maximum throughput for each storage pool
  - Specify how much is reserved for non-real-time I/O

```
Rtmb 200         # Max 200MBytes/s bandwidth on pool
RtmbReserve 10 # Reserve 10MBytes/s for non-RTIO
```

Affinities

- Use F_GETAFFINITY API call to get the affinity for a file
- Use F_SGINFO API call to query available storage pools
  - Each storage pool has an 8-byte key
- Use F_SETAFFINITY API call to set the affinity for a file

Example—Saving a File

Example—Saving a File

- open()—create a temporary file

Example—Saving a File

- open()—create a temporary file
- F_SETAFFINITY—optionally set the affinity of the file

Example—Saving a File

- open()—create a temporary file
- F_SETAFFINITY—optionally set the affinity of the file
- fcntl(F_PREALLOCATE)—preallocate the storage for the file

Example—Saving a File

- open()—create a temporary file
- F_SETAFFINITY—optionally set the affinity of the file
- fcntl(F_PREALLOCATE)—preallocate the storage for the file
- write()—write out the file's data

Example—Saving a File

- open()—create a temporary file
- F_SETAFFINITY—optionally set the affinity of the file
- fcntl(F_PREALLOCATE)—preallocate the storage for the file
- write()—write out the file's data
- close()—close the file

Example—Saving a File

- open()—create a temporary file
- F_SETAFFINITY—optionally set the affinity of the file
- fcntl(F_PREALLOCATE)—preallocate the storage for the file
- write()—write out the file's data
- close()—close the file
- rename()—rename the temporary file to the user-specified name

Developing for Xsan

- Don’t need to buy expensive SAN hardware to do basic development for Xsan
  - Desktop or PowerBook
  - FireWire drive
  - Network connection with static IP address
- Now you have a functional sandbox

Demo

Mike Margolis
“demo monkey”

http://developer.apple.com/wwdc2005/

Xsan-aware Application Examples

Xsan-aware Application Examples

- Backup software
  - Backup and restore affinity information along with other file attributes
  - Backup and restore quota information along with other volume attributes

Xsan-aware Application Examples

- Backup software
  - Backup and restore affinity information along with other file attributes
  - Backup and restore quota information along with other volume attributes
- Digital Asset Management
  - Use storage pools and affinities to isolate index, previews, and full-res data

Xsan-aware Application Examples

- Backup software
  - Backup and restore affinity information along with other file attributes
  - Backup and restore quota information along with other volume attributes
- Digital Asset Management
  - Use storage pools and affinities to isolate index, previews, and full-res data
- Distributed applications
  - Use Xsan to share data at high speeds

Xsan-aware Application Examples

- Backup software
  - Backup and restore affinity information along with other file attributes
  - Backup and restore quota information along with other volume attributes
- Digital Asset Management
  - Use storage pools and affinities to isolate index, previews, and full-res data
- Distributed applications
  - Use Xsan to share data at high speeds
- Streaming applications
  - Use extent preloading and bandwidth reservation

Xsan-aware Application Examples

- Backup software
  - Backup and restore affinity information along with other file attributes
  - Backup and restore quota information along with other volume attributes
- Digital Asset Management
  - Use storage pools and affinities to isolate index, previews, and full-res data
- Distributed applications
  - Use Xsan to share data at high speeds
- Streaming applications
  - Use extent preloading and bandwidth reservation
- Media ingest
  - Use preallocation and bandwidth reservation

Wrapup

- Xsan allows highly flexible use of storage
- Xsan allows shared access at high speeds
- Applications need to be aware of Xsan's characteristics
- Applications can be tuned for greater performance on Xsan
- It's easy to set up a sandbox to develop for Xsan

For More Information

- http://www.apple.com/xsan
- xsan-users@lists.apple.com

Documentation and Sample Code

Software Development Kit on Xsan CD

http://developer.apple.com/wwdc2005/

Q&A

Jason Thorpe
Xsan Engineering Manager
thorpej@apple.com

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference